[TC4] Probabilistic Generative Models - Lab 1

Mykola LIASHUHA, Vaibhav ARORA

1. Introduction

In this work, we implement a Variational Auto-encoder (VAE) with different distributions for latent and observed space. In particular, to deal with discrete latent space, we implement a gradient estimator known as Score Function Estimator (SFE). We also train a vanilla Autoencoder and turn it into a generative model by fitting a Gaussian-Misture Model (GMM) to the latent space from the encoder. We discuss VAEs, their implementation, and certain particularities owing to different types of observed and latent distributions, also comparing the to Sigmoid Belief Networks (SBNs).

2. Variational Auto-Encoders

VAEs essentially answer the question: How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? [1]. In our own terms, VAE is a type of generative model where we want to model the data distribution. Let f and g represent conditional distributions such that,

$$f : p(y|x), g : p(x|y)$$

If we are able to generate $x^{'}$ with g such that it is as close as possible to the input x, we can then claim that we have closely modeled the unknown distribution of the input data. f and g are taken to be neural networks since they are powerful function approximators. To generate $x^{'}$, we need to sample y but we cannot directly use $p(y|x)$ being conditioned on x. So we instead sample from a fixed known distribution (the prior) and model $f : p(y|x)$ (the posterior) to be as close to the prior. The training objective is the likelihood of the data $p(x)$. But since $f : p(y|x)$ is often intractable (required for the training objective) for any interesting models (requires marginalizing over the latent space), instead of working with it directly, we approximate it with a known family of distributions $q(y|x)$. Thus we end up having our objective to be a lower bound on the log-likelihood of the data which we try to maximize.

2.1. Variational Auto-Encoders in comparison to Sigmoid Belief Networks

Sigmoid Belief Networks (SBNs) are also designed to represent probability distributions over a set of attributes. The nodes in these networks represent random variables while the links represent causal influences. Typically trained using Contrastive Divergence with Gibbs Sampling procedures but with the Mean Field Theory [2] approach for SBNs, we also have a lower bound on the data likelihood as the training objective. So with the mean field method of SBNs, they are pretty much similar in the training objective. For each sample in the training set, we compute the associated bound on the log-likelihood and next adapt the weights of the belief networks by gradient ascent in the mean field bound [2]. For each unit, given an unbiased sample from the posterior distribution over the hidden states given the observed data, we maximize the log probability that its binary state in the sample from the posterior would be generated by sampled binary states of its parents. To do this, we cycle through the samples in the training set, maximizing their likelihoods for a fixed number of iterations or until one detects the onset of over-fitting, by updating the weights of the graph and the distribution parameters μ using mean field equations.

2.2. VAEs with different kind of distribution family

In general VAEs have a stochastic computation graph. The decoder needs a random sample from the latent space or from a distribution close to the latent space to generate samples. It is not possible to backpropagate through such a graph. Therefore we create surrogate functions which can be backpropagated through.

- For continuous latent space, for example, when our prior is a Gaussian and we train our encoder to learn the parameters of a Gaussian distribution, we use pathwise derivative estimators or commonly known as the reparameterization trick where we use the fact that any Gaussian can be rewritten as a linear function of a unit Gaussian and thus, the sampling procedure becomes an input to the network as opposed to being a part of the computation graph. We sample from the posterior $y\ q_\phi(y|x)$ using,

  $$y = g_\phi(x, \epsilon) = \mu + \sigma\epsilon$$

  where $\epsilon \sim (0, I)$

- For discrete latent space, such as Bernoulli, we get an estimator of the gradient via Score Function Estimator. The score function is a derivative of the log of a probability distribution with respect to its distributional parameters

  $$\nabla_\theta \log p(x;\theta) = \frac{\nabla_\theta p(x;\theta)}{p(x;\theta)}$$

  The general purpose estimator valid for both continuous and discrete variables is derived as follows:

  $$\nabla_\theta \mathbb{E}_{p(x;\theta)}[f(x)] = \nabla_\theta \int p(x;\theta) f(x) dx$$

  $$= \int f(x) \nabla_\theta p(x;\theta) dx$$

  $$= \int p(x;\theta) f(x) \nabla_\theta \log p(x;\theta) dx$$

  $$= \mathbb{E}_{p(x;\theta)}[f(x) \nabla_\theta \log p(x;\theta)]$$

  $$= \frac{1}{N} \sum_{n=1}^{N} f(x^{(n)}) \nabla_\theta \log p(x^{(n)};\theta); x^{(n)}\ p(x;\theta)$$

  One simple modification replaces the cost function with a shifted version of it,

  $$\nabla_\theta \mathbb{E}_{p(x;\theta)}[f(x)] = \mathbb{E}_{p(x;\theta)}[(f(x) - c) \nabla_\theta \log p(x;\theta)]$$

  where c is a constant referred to as a baseline. For any value of c, we still obtain an unbiased estimator because the additional term it introduces has zero expectation due to the property of the score. It allows for a simple but effective form of variance reduction [3]. This can be shown as follows:

  $$\nabla_\phi\ E_{q(y|x)}[log\ q(y|x)\ *\ (log\ p(x|y)\ -\ c(x))]$$

  $$= \nabla_\phi\ E_{q(y|x)}[log\ q(y|x)\ *\ log\ p(x|y)]\ -\ \nabla_\phi\ E_{q(y|x)}[log\ q(y|x)\ *\ c(x)]$$

  $$= E_{q(y|x)}[log\ p(x|y)\ \ *\ \nabla_\phi\ log\ q(y|x)\ ]\ \ -\ c(x) * \nabla_\phi \sum_y q(y|x)$$

  $$= E_{q(y|x)}[log\ p(x|y)\ \ *\ \nabla_\phi\ log\ q(y|x)\ ], since \sum_y q(y|x)\ =\ 1\ and\ \nabla_\phi \sum_y q(y|x\ =\ 0$$

- We let the observed space $p(x|z)$ be a multivariate Gaussian in case of real-valued data and Bernoulli in the case of binary.

The training objective is a lower bound on the log-likelihood of the data since it is intractable to compute directly, similar to SBN trained via mean field theory/variational methods.

3. Implementation

- How are VAE implemented? What should you take care about? What kind of distributions can be problematic?
Variational Auto-encoder consists of two parts that are similar to general Auto-encoder: encoder and decoder. The encoder is the first part which transfers the input data from its original space to latent space, and encoder is a basic deep neural network. But after this the difference of VAE and AE become obvious. After encoding the input into latent space, the output of encoder is no the encoded image itself, but actually encoder output values are parameters for the probabilities of the same type. So, the encoder transforms our input into distributions which we are going to use in order to sample from.

$$f(x;\varphi)$$

After forwarding input data through encoder, the distributions are going to be initialized with parameters. The parameters are actually the output values of encoder. For example, in case out latent distributions are Guassians then we have the latent space vector of size m and the size of encoder output would be m*num_parameters_per_distribution (for Gaussian distribution num_parameters_per_distribution = 2 or for Bernoulli num_parameters_per_distribution = 1). Having the distribution initialized, we sample from those distributions in order to get the actual latent space representation of size m.

$$y \sim q(y|x)$$

Also, in some cases with particular distributions we can also apply Reparametrization trick that was mentioned as one of the novel approaches in sampling the latent space representation. The main idea is to sample the prior distributions, and apply posterior distribution parameters to sampled prior data so that our final sampled values would follow the posterior distribution. Reparametrization tricks allows us to apply backward propagation in much faster way since without it we have to use Monte-Carlo approximation. It would be explaine later that we have to use Score Function Estimator in order to overcome limitations of MC approximation. It is important to note that this trick does not work fro Bernoulli distribution. In our work, we applied the trick with Gaussian latent space. Having obtained mu and sigma values (each one of size m) for posterior distribution, we sampled the m values from Normal distribution and applied the following formula:

$$y = e * \sigma + \mu,\ where\ \sigma,\ \mu\ are\ obtained\ from\ encoder\ output;\ e \sim\ N(0,1).$$

Having latent representation of our input data, we do forward propagation through our decoder. The decoder is also a representation in terms of distribution. So, the output values of decoder are the parameters for a corresponding distribution. For example, for gray-scale image we should use Bernoulli distribution since we only are looking for discrete representation values of our generated image. In case of Bernoulli, the output of decoder would be μ values.

After getting the output of decoder, we are going to maximize ELBO of VAE whereas ELBO itsself consists of several loss functions: reconstruction loss (BCE), Kullback–Leibler divergence (KL) and in some particular cases Score Function Estimate/Reinforce loss.

3.0.1 ELBO

The marginal likelihood is composed of a sum over the marginal likelihoods of individual datapoints which can be written as:

$$\log p_\theta(x) = KL(q_\phi(y|x)||p_\theta(y|x)) + \mathbb{L}(\theta, \phi; x)$$

As mentioned before, the posterior is intractable and we resort to learning known family of distributions and the first term on the right is the KL-divergence of the approximate posterior from the true posterior. Since it should always be greater than zero, the second term on the right side is called evidence lower bound or the marginal bound on the marginal likelihood of the data [1],

$$\log p_\theta(x) \geq \mathbb{L}(\theta, \phi; x) = \mathbb{E}_{q(y|x)}[-\log q_\phi(y|x) + logp_\theta(x, y)]$$

which can be written as

$$\mathbb{L}(\theta, \phi; x)\ = \mathbb{E}_{q(y|x)}[log\ p(x|y)]\ -\ KL(q_\phi(y|x)||p_\theta(z))$$

3.0.2 Reconstruction loss

The Reconstruction loss is used as a measurement of difference between original data and generated data, which was obtained by forwarding corresponding input data through encoder and decoder. This measurement criteria depends on the type of data of the original data: discrete or continuous. In our case our data are gray-scale image and dataset is discrete with 2 possible values: 0 and 1.

Since the dataset is binary then we would apply Binary-crossentropy loss as a core of reconstruction loss.

$$BCE(y,\ p)\ =\ -1\frac{1}{N}\sum_{i=1}^{N} y_i \times (log(p_i))\ +\ (1-y_i)\times(log(1-p_i))$$

Also, it is important to mention that the reconstruction loss equals to *minus* BCE loss: $log\ p(x|y) = -BCE$

3.0.3 KL Divergence

Kullback–Leibler divergence is a particluar mearement that shows a difference (not distance, since $KL[p|q]\ \neq\ KL\ [q|p]$) between two distirbutions of the same type. Therefore in our VAE we are going to apply KL loss function in order to understand how different prior and posterior latent distributions, are and afterwards we are going to lower the difference using this loss. This kind of minimization is applied in order to be able to generate new data points using the technique of sampling latent representation from prior distribution (due to the fact that after VAE training prior and posterior distributions are quite similar), and using obtained latent representation we generate a new data point.

3.0.4 Score Function Estimator/Reinforce

In our work we also applied Score Function Estimator in case of using Bernoulli distributions in latent space. Since we can not use reparameterization trick in case of discrete distribution, we apply Reinforce loss or also known as SFE as discussed above,

$$\mathbb{E}_{p(x;\theta)}[(f(x)-c)\nabla_\theta \log p(x;\theta)]$$

From implementation point of view, $f(x)$ is the reconstruction loss since it includes the data likelihood term and thus needs backpropagation over a stochastic nodes (not KL since it only requires the approximated posterior and prior). The computation is done in line with the equation above. The baseline term is taken to be the running average of the reconstruction loss. One practical trivia that the reconstruction term needs to be detached while computing the baseline term.

4. Results

4.0.1 Gaussian VAE

In second part of the practical exercise we implemented Variational AutoEncoder with Gaussian latent space. In this model we applied Gaussian distribution during latent space sampling. In our best model we used such parameters: - epochs: 20,
- latent dim: 2,
- hidden dim: 400

The final loss function looks quite simple due to Reparametrization trick:

$$Loss = -BCE - KL$$

Below we are going to provide a visual pipeline how the training is conducted:

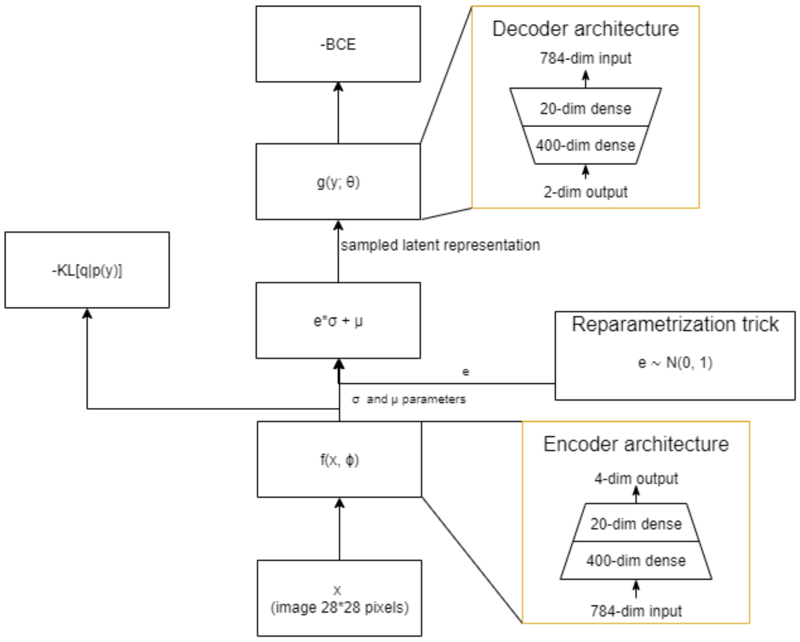
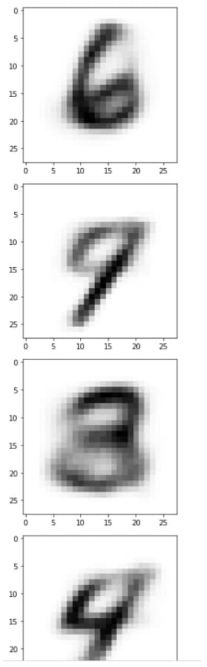
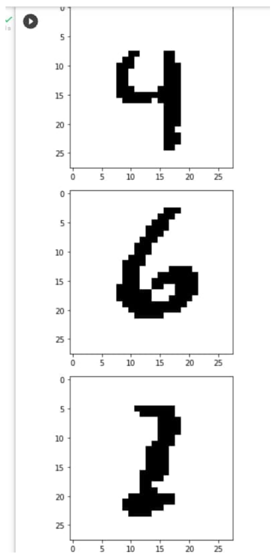


Figure 1: Gaussian VAE computation pipeline



(a) Generated distributions



(b) Sample images from generated distributions

Figure 2: Generated data

4.0.2 Bernoulli VAE

In third part of the practical exercise we implemented Variational AutoEncoder with Bernoulli latent space. In this model we applied Gaussian distribution during latent space sampling. In our best model we used such parameters: - epochs: 50,

- latent dim: 20,
- hidden dim: 400

The final loss function in this case is more complicated due to complications of MC approximation:

$$Loss = -BCE + SFE - KL,$$

$$where\ SFE = latent\_space\_log\_probabilities * (-BCE - c(x))$$

Below we are going to provide a visual pipeline how the training is conducted:

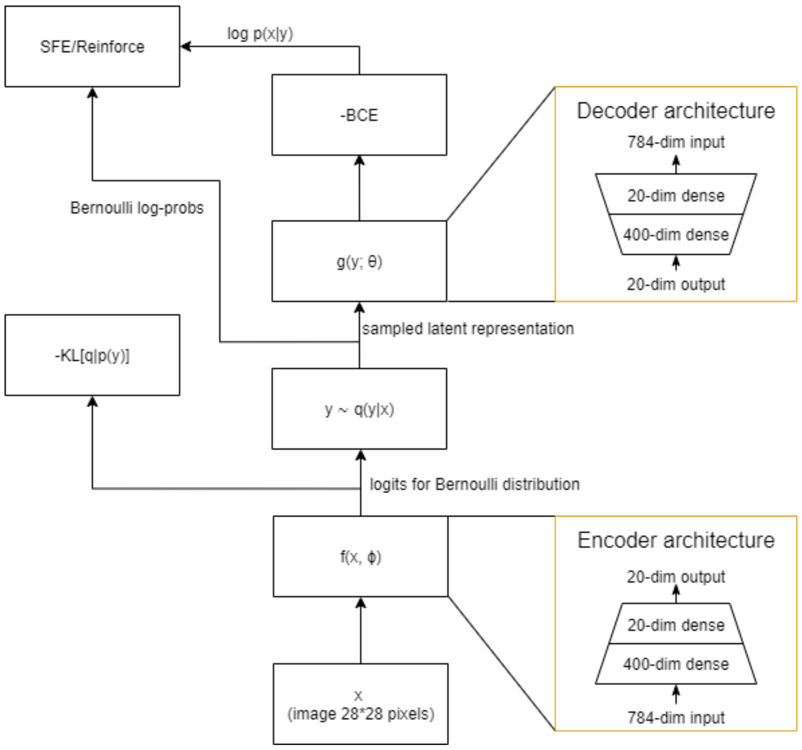
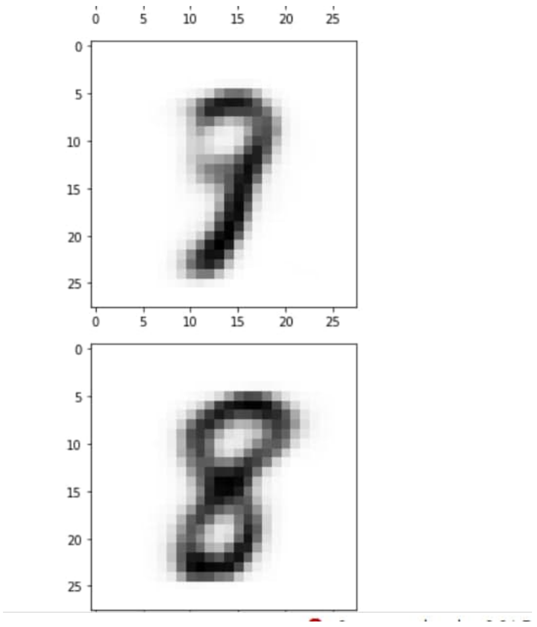


Figure 3: Bernoulli VAE computation pipeline


(a) 50 epochs

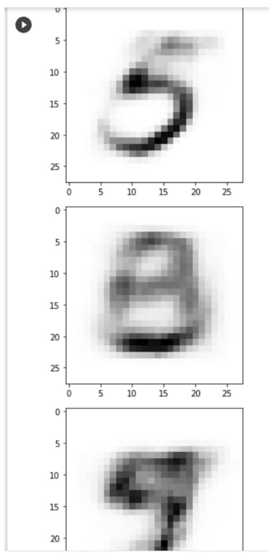

(b) 30 epochs

Figure 4: Generated images

4.0.3 GMM AE

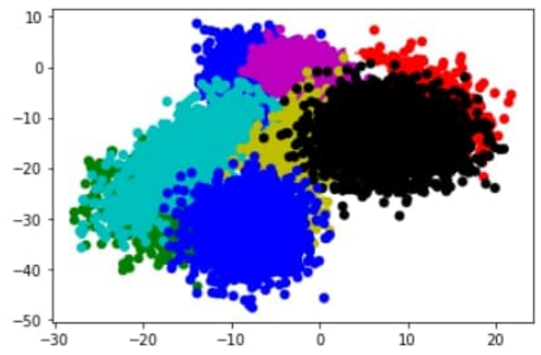


Figure 5: 9 clusters of outputs of latent size 2

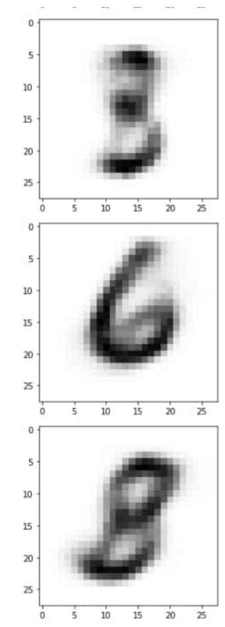


Figure 6: Generated samples GMM

References

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014. 1, 3

[2] M. I. Jordan L. K. Saul, T. Jaakkola. Mean field theory for sigmoid belief networks, 1992. 1

[3] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning, 2020. 2